

**CLAIMS**

We claim:

1. A method for processing queries of hierarchical tagged data using hints, said hints being navigational aids and said processing being performed on a computing device,
  - providing a plurality of hints for the hierarchical tagged data, said data having a plurality of nodes  $l$  and  $c$  such that  $l$  is a parent of  $c$ ;
  - pruning said plurality of hints to avoid unnecessary navigation when processing said queries;
  - updating said hints in accordance with required navigation workload and updates and changes to the hierarchical tagged data; and
  - selecting techniques for hints according to limitations on an allocated memory size of said computing device.
2. The method of claim 1, wherein the hint being represented as  $h(l, c, t)$ , where  $t$  is a tag of a child node accessible in top-down traversal from  $c$ , said hint being positive if  $t$  exists and otherwise negative.
3. The method of claim 1, further comprising the steps of:
  - matching hint information at a currently accessed node  $n$  with a remaining query path  $q$ ;
  - analyzing all hints where  $c$  is a child of node  $n$ ; and
  - eliminating from query processing a sub tree rooted at each child  $c$  of node  $n$  having a tag  $t$ .
4. The method of claim 1, further comprising the steps of:
  - a) for every query path  $q$ , identifying all children  $c$  of a current node  $n$  having a tag  $t$  to be visited in a next step of query processing;
  - b) for each tag  $t$  to match in said query path  $q$ , determining all hints such that  $c$  is a child of  $n$ ;
  - c) eliminating from query all said children  $c$  of said current node  $n$  having said tag  $t$  to be visited in said next step of query processing;

d) determining a query constraints and further reducing said children  $c$  having said tag  $t$  to be visited in said next step of query processing in accordance with said constraints;

e) for each said child  $c$  having said tag  $t$ , setting sub queries  $q'$  corresponding to a sub tree rooted at said child  $c$  having said tag  $t$ , and

f) repeating steps (a) through (e).

5. A method of utilizing one or more hints for query processing over a hierarchical tagged data structure in a computing system having memory, the data structure having a plurality of nodes  $l$  and  $c$  such that  $l$  is a parent of  $c$ , the hint, represented as  $h(l, c, t)$ , being positive if there is a tag  $t$  accessible in top-down traversal from  $c$  and otherwise negative, said method comprising steps of:

for each tag in the XML document

calculating each hint and a usefulness of each hint;

selecting a number of hints  $k$  having a greatest usefulness, where  $k$  equals a total memory size divided by a size of the hint; and

eliminating redundant hints.

6. The method of claim 5, further providing a usefulness matrix for calculating said usefulness of each of said hints, wherein for a pre-defined parameter  $0 \leq \alpha \leq 1$ ,

the usefulness of the hint is calculated as  $Usf_{h(l,c,t)} = (1 + \alpha \times semW_{h(l,c,t)}) \times sUsf_{h(l,c,t)}$ , where  $semW_{h(l,c,t)}$  is a semantic weight and  $sUsf_{h(l,c,t)}$  is a structural usefulness of the hint.

7. The method of claim 6, wherein said structural usefulness of a hint is a number of nodes of said data structure that can be pruned out the search space for a query “// $t$ ” if the hint is materialized.

8. The method of claim 5, wherein only negative hints are used.

9. A method of utilizing one or more hints for query processing over a hierarchical tagged data structure in a computing system having memory, the data structure having a plurality of nodes, the hint being positive if there is a tag  $t$  accessible in top-down traversal from a child node and otherwise negative, said method comprising steps of:

for each tag in the data structure:

- (a) calculating a bitmap for a current node  $B(current)$  with all bits set to one;
- (b) setting a bit of a current tag  $B(current)[tag(current-tag)]$  to zero;
- (c) calculating a plurality of possible non-redundant hints for each child node; and
- (d) refreshing a hint list.

10. The method of claim 9, wherein step (a) further comprises the steps of:  
 calculating a bitmap for each child node of said current node;  
 AND-ing all said bitmaps for each child node; and  
 setting a bit corresponding to tag ID  $B(current)[tagid(current - tag)]$  of a current tag to zero if said current tag exists.

11. The method of claim 9, wherein step (c) further comprises the steps of:  
 for each bit  $j$  such that  $B(current)[j]$  is equal to zero and  $B(child)[j]$  is equal to one:  
 (c1) determining if there is a need to add a hint  $h(current\ node, current\ child, tag(j))$  to a list of hints;  
 (c2) eliminating a least useful hint from said list if said list is full; and  
 (c3) adding said hint to said list.

12. The method of claim 11, wherein step (c1) further comprises the step of determining if a usefulness value  $Usf[h(current\ node, current\ child, tag(j))]$  of said hint is greater than the least useful hint in said list.

13. The method of claim 9, wherein only negative hints are used.

14. A computer program device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for utilizing one or more hints for query processing over a hierarchical tagged data structure in a computing system having memory, the data structure having a plurality of nodes, the hint being positive if there is a tag accessible in top-down traversal from a child node, and otherwise negative, said method comprising steps of:

for each tag in the data structure:

- (a) calculating a bitmap for a current node  $B(current)$  with all bits set to 1;
- (b) setting a bit of a current tag  $B(current)[tag(current-tag)]$  to zero;

- (c) calculating a plurality of possible non-redundant hints for each child node; and
- (d) refreshing a hint list.

15. The method of claim 14, wherein step (a) further comprises the steps of:  
 calculating a bitmap for each child node of said current node;  
 AND-ing all said bitmaps for each child node; and  
 setting a bit corresponding to tag ID  $B(current)[tagid(current - tag)]$  of a current tag to zero if said current tag exists.

16. The method of claim 14, wherein step (c) further comprises the steps of:  
 for each bit  $j$  such that  $B(current)[j]$  is equal to zero and  $B(child)[j]$  is equal to one  
 (c1) determining if there is a need to add a hint  $h(current\ node, current\ child, tag(j))$  to a list of hints;  
 (c2) eliminating a least useful hint from said list if said list is full; and  
 (c3) adding said hint to said list.

17. The method of claim 16, wherein step (c1) further comprises the step of determining if a usefulness value  $Usf[h(current\ node, current\ child, tag(j))]$  of said hint is greater than the least useful hint in said list.

18. The method of claim 15, wherein only negative hints are used.